UCRL- 93546
PREPRINT

A Low-Cost General Purpose Spectral Display Unit Using an IBM PC

Sandra L. Robinson

IEEE Transactions on Nuclear Science
1985 Nuclear Science Symposium
San Francisco, CA
October 23, 1985

October 1985

Lawrence Livermore National Laboratory

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

A LOW-COST GENERAL PURPOSE SPECTRAL DISPLAY UNIT USING AN IBM PC*

Sandra L. Robinson
Lawrence Livermore National Laboratory
P.O. Box 808
Livermore, California 94550

Abstract

Many physics experiments require acquisition and analysis of spectral data. Commercial minicomputer-based multichannel analyzers collect detected counts at various energies, create a histogram of the counts in memory, and display the resultant spectra. They acquire data and provide the user-to-display interface. The system discussed separates functions into the three modular components of data acquisition, storage, and display. This decoupling of functions allows the experimenter to use any number of detectors for data collection before forwarding up to 64 spectra to the display unit, thereby increasing data throughput over that available with commercial systems. An IBM PC was chosen for the low-cost, general purpose display unit. Up to four spectra may be displayed simultaneously in different colors. The histogram saves 1024 channels per detector, 640 of which may be distinctly displayed per spectra. The IEEE-488 standard provides the data path between the IBM PC and the data collection unit. Data is sent to the PC under interrupt control, using direct memory access. Display manipulations available via keyboard are also discussed.

Introduction and Motivation

Many types of physics experiments require acquisition and analysis of spectral data. Currently, minicomputer-based multichannel analyzers may be purchased to collect detected counts at various energy levels, create a histogram of the counts in computer memory, and display the resultant spectra. They handle data acquisition and also interface to the operator for manipulating the display. The number of detectors in use at the same time is limited to the number of physical connectors on the unit. Memory for saving events competes with memory used for data aoquisition and the user interface.

Decoupling the functions in the system would remove current limitations by allowing the experimenter to customize his or her system for flexible experimentation. Consider the system as divided into three blocks. One block consists of data acquisition equipment, a second provides data storage and forwarding, while a third handles real-time display of the data forwarded. Since the display system can concentrate on display functions only, most of memory may be devoted to real-time event storage for display. Data may be sent to the display unit asynchronously, encoded with the identification of the detector involved. Data throughput is no longer limited to the number of physical connectors on the display unit. Modularity also leads to easier maintainability. The use of standard components minimizes maintenance costs and ensures a plug-in capability for a variety of experimental set-ups.

The requirements of a low cost, general purpose, real-time display system were determined. These included an inter-computer communications link handled in real-time (under direct memory access and interrupt control), a histogram of the data, the user interface, and the actual display. Up to four spectra may be displayed simultaneously in different colors. Techniques for spectral expansion and manipulation were designed and implemented. This paper describes the system environment, components, and the system's capabilities. Specifically described are the communication, display, and the user interface subsystems.

Environment

Figure 1 shows the display system's external environment. The multiple alpha detector (MAD) system is shown with 32 detector pairs for a total of sixty four (64) detectors. This particular detector system is being used to investigate the alpha and spontaneous fission activities of short-lived transfermium isotopes by the real-time monitoring of the mother-daughter alpha decay sequence [1]. A high speed Computer Automated Measurement and Control (CAMAC) system acquires data from the detectors. The data is stored by an LSI-11/23 minicomputer which forwards the spectral data to an IBM personal computer, the display unit used for the project.

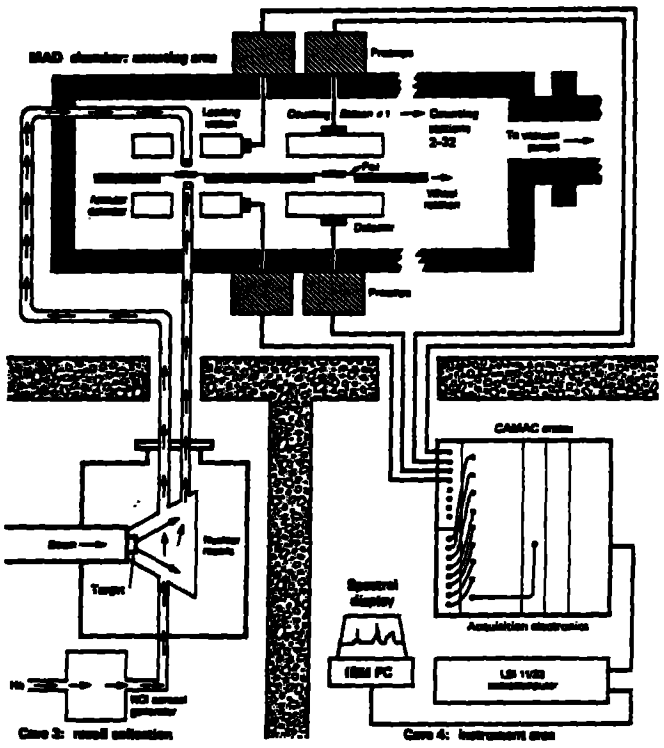

Figure 1. Multiple Alpha Detector System

System Components

The display system is shown in Figure 2. The personal computer has a total available memory of 512K bytes, two floppy disk drives, an IBM color

* Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract number W-7405-ENG-48.

display monitor, keyboard, graphics printer, color graphics display card, and an IEEE-488 standard interface card for receiving data from the LSI-11.

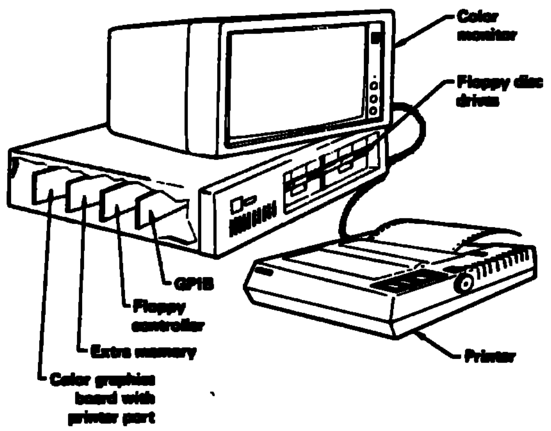

Figure 2. Display Components

A number of personal computers were considered for use as the display unit. The current market was a major driving factor in the choice of the IBM PC. While not very innovative as a computer, the IBM PC has invaded a wide range of markets. This has occurred because of the IBM name, the availability and reliability of parts and software, and the resulting standardization. IBM encouraged third parties to make compatible peripherals, memory, and software [2].

Software was written in both assembly language and Fortran. Assembly was necessary for many of the real-time functions. Fortran was chosen primarily because it was the standard for the intended users of the system.

Computer Communications

Providing a real-time link between the LSI-11 and the IBM PC was perhaps the most difficult task in the project. At the time of design, little to nothing existed in the way of complete and adequate technical documentation on the personal computer. The following description includes the direct memory access (DMA) link using interrupts on the PC, and the histogram created as a result of the interrupts.

In a real-time experiment, it is unacceptable to require the master processor (the LSI in this case) to wait for the slave (the personal computer) to accept data. Current General Purpose Interface Buses (GPIBs, which follow the IEEE 488 standard) allow data transfer between an LSI-11 and an IBM personal computer [3]. However, available software required the slave to request to read data. The project requirements dictated a true direct memory access (DMA) to the PC, whereby the data is automatically transferred to PC memory without software polling and read commands. A low level software driver was created for this purpose. Three chips required programming. These were the GPIB controller (7210), DMA controller (8237), and the interrupt controller (8259A). The latter two are on the PC system board [4]. The GPIB controller is on a board installed in the PC. The GPIB controller on the sending end is also installed in the LSI-11. Data may be stored for further analysis at the LSI-11 before being forwarded to the PC. Simple commands transport data packets from the LSI to the PC. However, once sent, data is automatically accepted in the PC memory. This new approach to inter-computer communication with the PC has yielded a much faster, truly real-time data transfer. The result is that the personal computer is a viable instrument in the laboratory for data display.

The GPIB board at the LSI-11 side is capable of providing data speeds up to 250K bytes per second between the IEEE-488 bus and the DEC LSI-11 bus. It provides the hardware for decoding GPIB commands and implements all functions defined in the IEEE-488 specifications. It is programmed in this system to be the Controller and talker, and can send interface messages to the IBM PC. The driver program is written in Fortran under the RT-11 operating system. The program resets the GPIB and sets up the talker and listener addresses. It then reads from available raw data files, preprocesses them by pulling out spectral data, and places the data in packets of 2048 bytes. When a packet is full, the code sends the packet over the IEEE-488 bus to the PC. It waits to receive an acknowledgement from the PC before assembling the next packet.

All initialization code on the PC of the GPIB, DMA, and interrupt controllers is written in assembly language. The GPIB interface on the PC end is initialized to the correct listen address. The chip is initialized to listen mode. DMA is enabled, and the interrupt is set up to interrupt on error. The DMA controller chip on the PC system board keeps one DMA channel open for reading data. The receive buffer address and size are initialized. Now the data may be transferred over the IEEE-488 bus and immediately written into the receive buffer. The remaining link is the interrupt controller.

The 8088 microprocessor on the PC system board can respond to three kinds of interrupts: software, nonmaskable external, and maskable external. Critical events such as impending power failures cause nonmaskable interrupts. The maskable interrupt is used here when the receive buffer fills. The PC has eight maskable interrupt lines, three of which it uses for the system timer, keyboard, and diskette adapter. The interrupt controller must be initialized to one of the remaining request lines. When the receive buffer is full, an interrupt request signal is sent to the interrupt controller. The controller checks to see if the interrupt is masked. If not, it alerts the 8088 of the interrupt. The 8088 acknowledges the interrupt. The 8259 controller releases a pre-initialized base with which the 8088 calculates the vector address for the interrupt service routine [5]. The latter will check the status of the GPIB board to see if an error occurred. If not, the software knows that the receive buffer filled and generated the interrupt. (There is no direct way to interrupt on a full buffer. That is why the error mode is chosen for an interrupt and the error status must subsequently be checked to see if it was, indeed an error interrupt as opposed to a filled buffer.) Reading the status automatically clears the interrupt line [6]. The service routine temporarily disables interrupts and DMA while it addresses an alternate buffer for receiving the next packet. This double buffering is necessary to allow the received data to be histogrammed while collecting new data.

The standard operating mode of the 8259 is called the fully nested mode. This means that all interrupt request channels are priority ordered in a circular fashion. The highest priority channel is serviced first, and then becomes the lowest priority. This ensures that all interrupts are serviced. Priority modes can be changed by programming the 8259 chip, but the standard mode was used for this project.

One of the more challenging tasks in the project was to ensure compatibility between the Fortran and Macro assembler compilers. Typically, on the PC, assembly code is used for small functions requiring little data space. This project needed a histogram size of over 200K bytes. The 8088 operates on segments which are 64K bytes in size. This meant that separate data segments had to be specified with careful attention given to preserve the Fortran environment when exiting to the assembly environment. The first Fortran compiler available did not allow specification of these extra data segments directly. It would place the stack in the middle of these segments, ignoring their definition in the assembly code. Fortunately, a new version of the compiler was released mid-project which corrected the problem. Segment pointers are saved upon entering the assembly code, new segment pointers are set up, and old ones are restored before return to the Fortran main code. This eliminates conflict between the two languages.

The histogram is developed in assembly. Three eight bit bytes are used for each count. Screen updates are made via Fortran graphics library calls. Therefore, the histogram routine saves a copy of the data along with its current count in the histogram for the Fortran code to use for screen updates. This way, a mirror image is made of the receive buffer along with current relevant counts in memory, the receive buffer may be used for more data, and the histogram may be updated without waiting for the Fortran screen update. A data word is 16 bits long. The upper 6 bits identify the detector involved, and the lower 10 bits contain the channel number. These bits are used to determine the address in the histogram for incrementing new counts.

A word is needed here concerning the use of the two languages for this project. The Fortran compiler used did not allow for byte manipulation, nor could one address absolute memory locations directly. For speed, the interrupt routines and histogram formation were written in assembly. The main code was written in Fortran because it was the language of choice for the customer. The screen updates were called from Fortran largely to maintain a standard calling format so that the user could add graphics as needed, and so that replacement of the graphics board would be more direct. The latter quickly became an evident need, as is explained below.

Display

The color graphics board and monitor used provide 640 X 200 pixel resolution with a palette of 16 colors. No higher resolution is required for the application, and the cost may thus be kept to a minimum. The board contains its own refresh memory, video processing, timing and control, and parallel interface logic. The board came with a complete library of basic graphics programs, callable in Fortran, Basic, assembly or Pascal. The commands used were simply a "draw line" for drawing the cursors, and "put point" for plotting the histogram [7]. Figure 3 shows a typical display of data. This board needs to be replaced in the future because the manufacturer went out of business. Fortunately, it is an IBM compatible board with standard graphics library calls, and the switchover should be fairly simple.

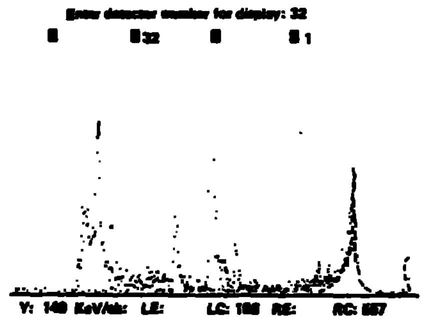

Figure 3. Displayed Spectra

User Interface

Figure 4 shows the keyboard layout. Starting with the function keys, F1-F4 are selection keys used to choose spectra to display. The "erase 64" key zeroes the spectra in the PC memory, as well as on the screen if any are currently displayed. The "erase sums" key zeroes the four spectral sums in memory, along with any summed spectra on display. The "domain" key cycles around the selected specta, one spectra per toggle of the key. This selects the current domain, which defines the color of the cursors to match the current spectra, and it defines the spectra with which cursor movement and baseline movement are defined. The baseline key will toggle from the common baseline to a pre-defined baseline for the current spectra. The last two function keys provide x and y axis expansion and contraction. The y scale moves from a linear scale to a log scale and back again. The "Alt" key is held for expansion along with the function key. The function key pressed alone will contract the display.

The upper keypad contains the cursor movement keys, four keys for two cursors moving in two directions each. Region keys allow region definition between cursors, region advancement, deletion, and expansion of the region area to fill the screen. Regions may also be summed. A calibration key allows the operator to define keV

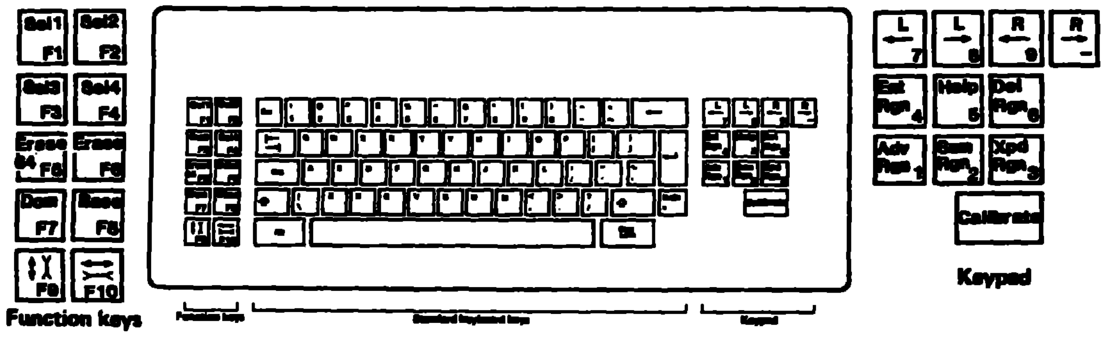

Figure 4. Keyboard Layout

per channel. A "help" key gives the operator a picture of the keyboard layout, with definitions of available functions.

The Fortran main program polls an assembly keyboard routine to see if a key has been pressed. This routine calls a keyboard interrupt in the BIOS, which is the program area that controls standard input and output functions in the PC. These are in ROM. The interrupt routine returns a key number if one was pressed, and the main program calls the appropriate completion routine.

System Limitations and Recommendations

As mentioned earlier, the color graphics board needs to be replaced with a currently manufactured board. At the same time, screen updates should follow histogram updates on a point by point, rather than packet by packet basis. This would prevent having to make a copy of the state of the histogram for the Fortran screen update. The lack of compatibility between Fortran and assembly, as well as the lack of desirable functions in the Fortran compiler dictated the current design. Perhaps a newer compiler version will correct current deficiencies. For increased speed, screen updates could directly address graphics memory, as opposed to the current method of using library calls. The latter was done to make the simplest and most straightforward interface to the graphics board. This aids upgrading or replacing the graphics board.

A language such as C would be preferable to Fortran if the user community were comfortable with it. C would interface nicely to the assembly portions of the code. This alone would have saved much development time.

Ideally, the display system would be able to keep up with the fast data transfer rate. However, the processor on the personal computer is limited in speed. Also, the screen update is limited by a relatively slow retrace signal.

A number of things should be done to speed up execution of the code. One immediate way to do this would be to upgrade the PC from an 8-bit data path to a 16-bit path using one of the recent boards on the market made for this purpose. This was tested on the current system and showed a data throughput rate increase of 260%.

Functionally, before a new point is painted on the screen, the last point painted must be erased. Unfortunately, if two colors cross, a color may be erased, leaving "holes" in its spectra. This could be eliminated with more sophisticated hardware or software. This, however, increases the cost of the graphics. Currently, the operator may request that a spectra be redrawn.

Summary

The IBM PC may be successfully used as a low cost alternative for spectral display. In its current environment,it behaves as a modular display component in a versatile experimental system. It can support up to 64 different detectors, displaying four spectra simultaneously in different colors. It may be attached via a standard GPIB bus to an LSI-11. Data is sent across the bus and placed directly in the personal computer's memory without the need for polling or software requests. Of the 1024 channels saved in memory, 640 of them may be distinctly displayed. Two cursors track the current spectra. A variety of functions are available at the keyboard for spectral manipulation. These include erasing spectra, changing baselines, expanding and contracting the x and y axes, cursor movement, defining regions between the cursors, summing regions, and expanding regions to fill the screen.

On commercial systems, [8] data acquisition as well as display are in one unit. This limits the number of detectors in use to the number of physical connections on the acquisition/display unit (typically four). The personal computer is dedicated to display. It is a user-friendly, low-cost system. Use of standardized components ensures a plug-in capability with other computers connected using an IEEE 488 standard interface. Four spectra displayed simultaneously in different colors add to the versatility of the display system for real time spectral display.

As faster graphics controllers and add-on boards are developed, the concept of using a personal computer for scientific applications will become more viable. The current configuration is fast enough for some applications, but would not be able to handle data rates above 2500 bytes per second. Future developments on the horizon for personal computers promise to provide greater speed and more options. This will better support the use of these computers for scientific and engineering purposes, as opposed to the business community for which the market is currently geared.

References

[1] E. Watkins, R. Dougan, J. McQuaid, "A LSI/CAMAC System for Heavy Elements Research." IEEE Transactions on Nuclear Science, NS-32:4 (August 1985), 1453-56.

[2] J. Pingry, "The Expanding Realm of the IBM PC." Digital Design, (February 1984), 80-86.

[3] GPIB-PC User Reference Manual For the IBM Personal Computer. National Instruments, (September 1983).

[4] iAPX 86,88 User's Manual. Intel Corporation, (August 1981), A135-A174.

[5] C. Dunford, "Interrupts and the IBM PC, Part I." PC Tech Journal, 1:3 (November/December 1983), 173-199.

[6] C. Dunford, "Interrupts and the IBM PC, Part II." PC Tech Journal, 1:4 (January 1984), 144-186.

[7] MicroGraphics System Operation and Reference Manual. MicroGraphics Technology Corporation, 1983.

[8] ND66 Multichannel Analyzer/Remote Terminal Operator's Instruction Manual. Nuclear Data, Inc., 1981.

[9] M. Franklin, Using the IBM PC: Organization and Assembly Language Programming. Holt, Rinehart and Winston, 1984.

[10] IBM Technical Reference. IBM Corporation.

[11] L. J. Scanlon, "Mining the System Resources." PC World, (November 1983), 230-241.